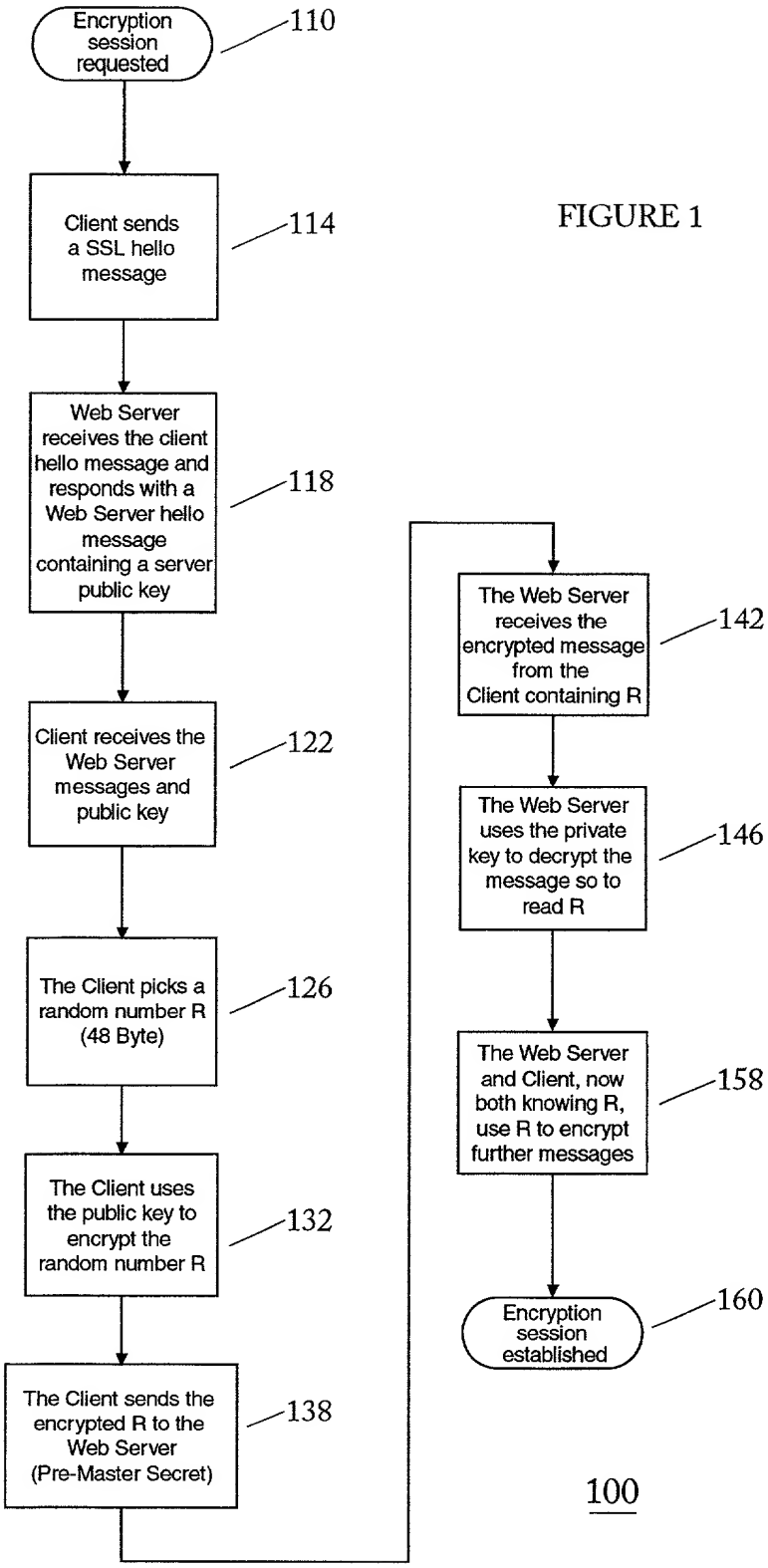

Encryption session requested
110
Client sends a SSL hello message
114
Web Server receives the client hello message and responds with a Web Server hello message containing a server public key
118
Client receives the Web Server messages and public key
122
The Client picks a random number R (48 Byte)
126
The Client uses the public key to encrypt the random number R
132
The Client sends the encrypted R to the Web Server (Pre-Master Secret)
138
The Web Server receives the encrypted message from the Client containing R
142
The Web Server uses the private key to decrypt the message so to read R
146
The Web Server and Client, now both knowing R, use R to encrypt further messages
158
Encryption session established
160
100

FIGURE 1

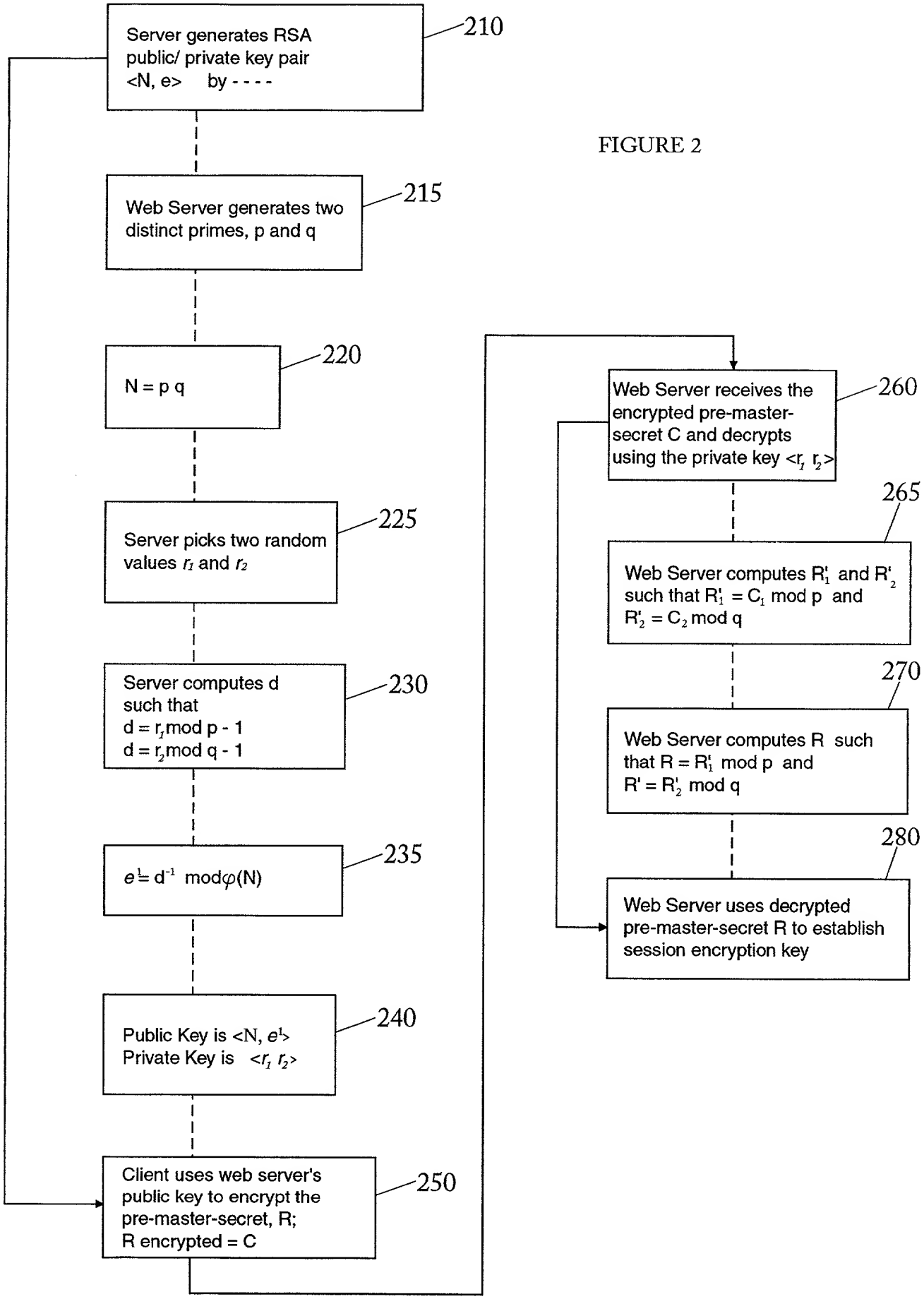
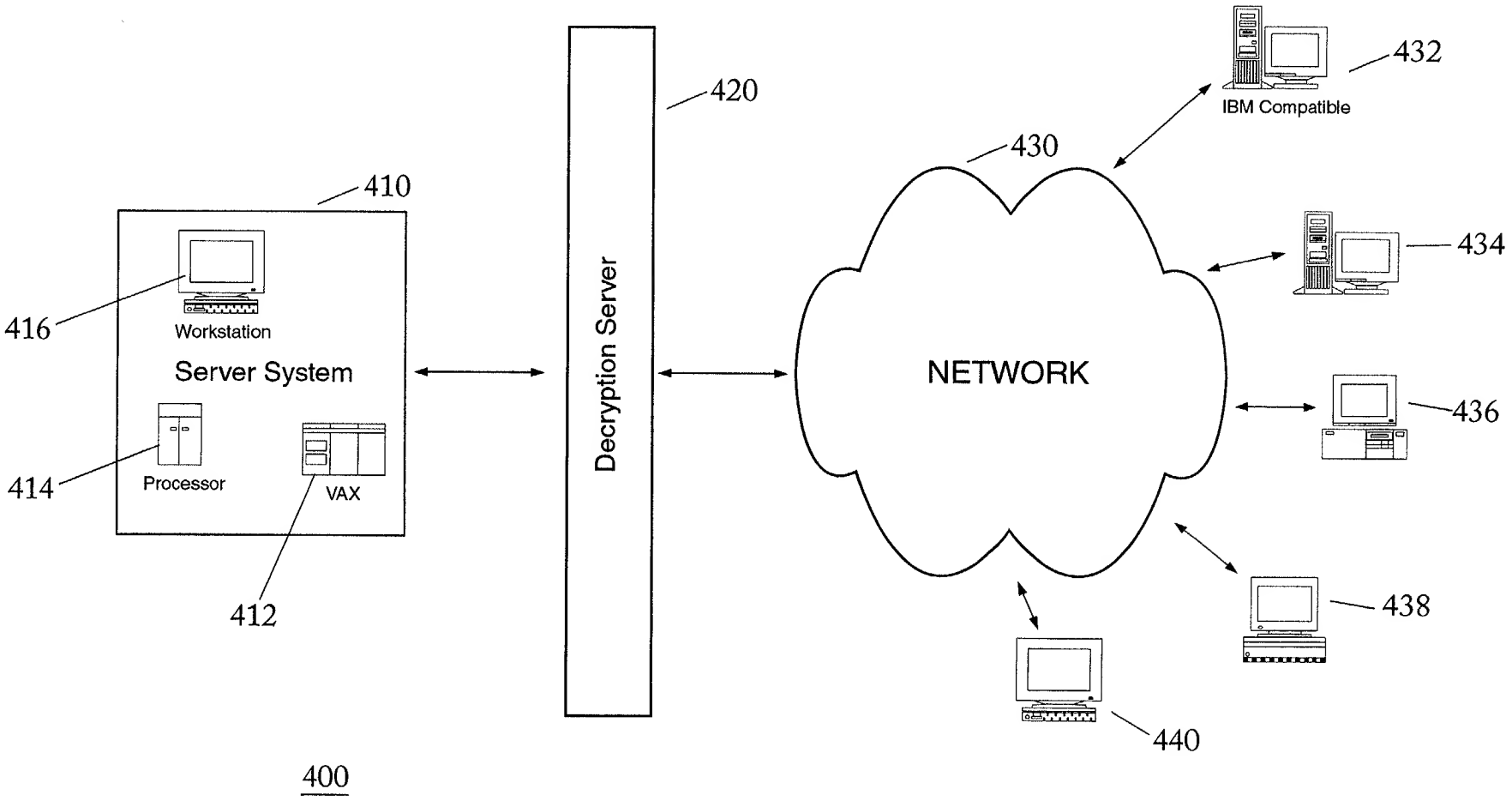

Server generates RSA public/ private key pair <N, e> by - - - -
210
Web Server generates two distinct primes, p and q
215
N = p q
220
Server picks two random values r_1 and r_2
225
Server computes d such that
$d = r_1 \bmod p - 1$
$d = r_2 \bmod q - 1$
230
$e^{-1} = d^{-1} \bmod \varphi(N)$
235
Public Key is <N, e^{-1}>
Private Key is <r_1 r_2>
240
Client uses web server's public key to encrypt the pre-master-secret, R;
R encrypted = C
250
Web Server receives the encrypted pre-master-secret C and decrypts using the private key <r_1 r_2>
260
Web Server computes R'_1 and R'_2 such that $R'_1 = C_1 \bmod p$ and $R'_2 = C_2 \bmod q$
265
Web Server computes R such that $R = R'_1 \bmod p$ and $R' = R'_2 \bmod q$
270
Web Server uses decrypted pre-master-secret R to establish session encryption key
280

FIGURE 2

FIGURE 4


410
416
Workstation
Server System
414
Processor
VAX
412
420
Decryption Server
430
NETWORK
432
IBM Compatible
434
436
438
440

400

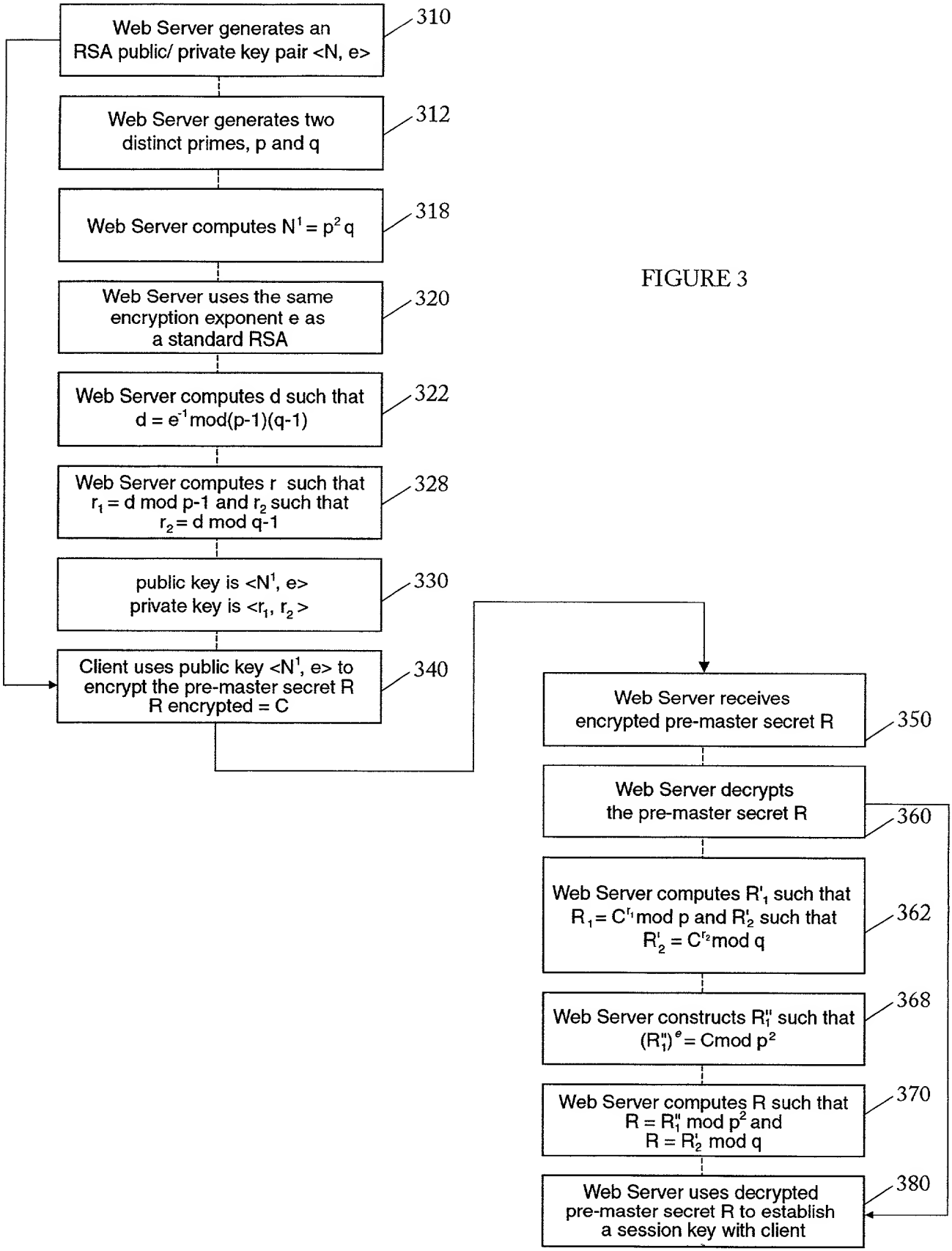

Web Server generates an RSA public/ private key pair <N, e>
310
Web Server generates two distinct primes, p and q
312
Web Server computes $N^1 = p^2 q$
318
Web Server uses the same encryption exponent e as a standard RSA
320
Web Server computes d such that $d = e^{-1} \bmod (p-1)(q-1)$
322
Web Server computes r such that $r_1 = d \bmod p-1$ and r_2 such that $r_2 = d \bmod q-1$
328
public key is $<N^1, e>$ private key is $<r_1, r_2>$
330
Client uses public key $<N^1, e>$ to encrypt the pre-master secret R R encrypted = C
340
Web Server receives encrypted pre-master secret R
350
Web Server decrypts the pre-master secret R
360
Web Server computes R'_1 such that $R_1 = C^{r_1} \bmod p$ and R'_2 such that $R'_2 = C^{r_2} \bmod q$
362
Web Server constructs R''_1 such that $(R''_1)^e = C \bmod p^2$
368
Web Server computes R such that $R = R''_1 \bmod p^2$ and $R = R'_2 \bmod q$
370
Web Server uses decrypted pre-master secret R to establish a session key with client
380

FIGURE 3

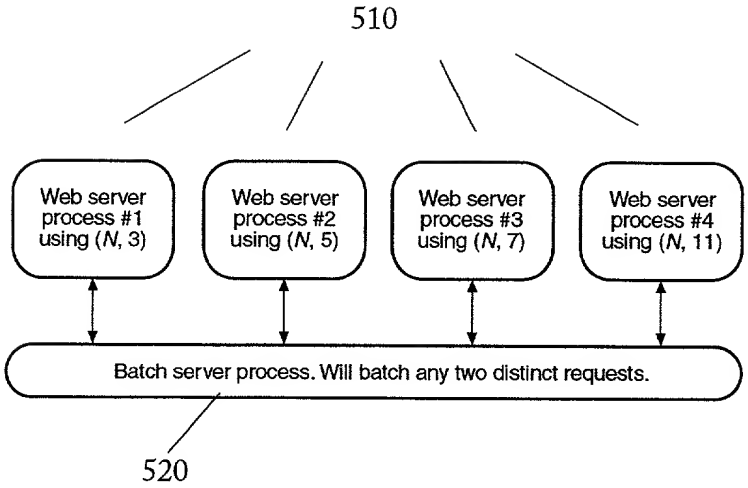
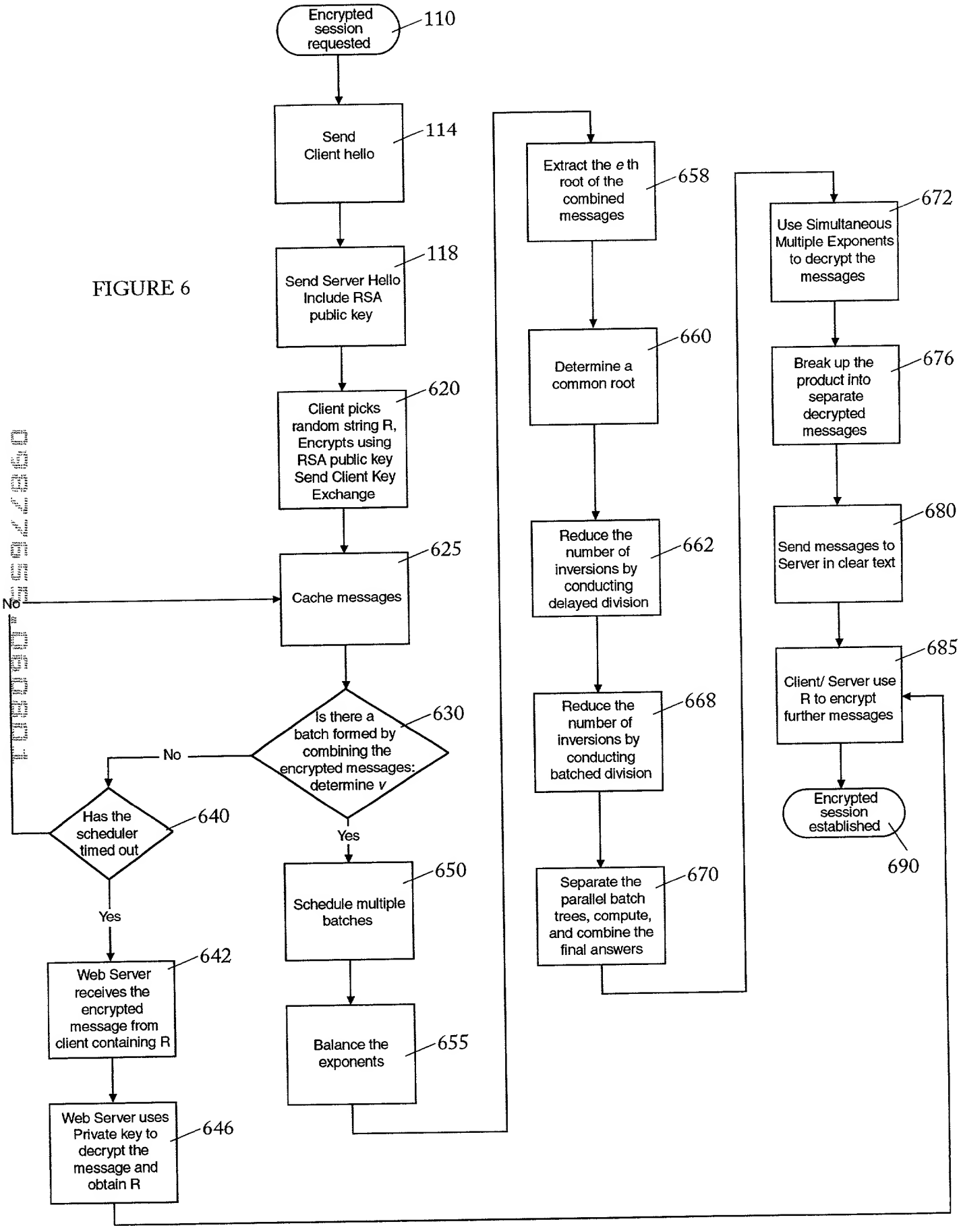

510
Web server process #1 using (N, 3)
Web server process #2 using (N, 5)
Web server process #3 using (N, 7)
Web server process #4 using (N, 11)
Batch server process. Will batch any two distinct requests.
520

FIGURE 5

FIGURE 6


Encrypted session requested
110
Send Client hello
114
Send Server Hello Include RSA public key
118
Client picks random string R, Encrypts using RSA public key Send Client Key Exchange
620
Cache messages
625
Is there a batch formed by combining the encrypted messages: determine v
630
No
Yes
Has the scheduler timed out
640
No
Yes
Schedule multiple batches
650
Balance the exponents
655
Extract the e th root of the combined messages
658
Determine a common root
660
Reduce the number of inversions by conducting delayed division
662
Reduce the number of inversions by conducting batched division
668
Separate the parallel batch trees, compute, and combine the final answers
670
Use Simultaneous Multiple Exponents to decrypt the messages
672
Break up the product into separate decrypted messages
676
Send messages to Server in clear text
680
Client/ Server use R to encrypt further messages
685
Encrypted session established
690
Web Server receives the encrypted message from client containing R
642
Web Server uses Private key to decrypt the message and obtain R
646